Development of MD Engine: High-Speed Accelerator with Parallel Processor Design for Molecular Dynamics Simulations

SHINJIRO TOYODA,[1] HIROH MIYAGAWA,[2] KUNIHIRO KITAMURA,[2] TAKASHI AMISAKI,[3] EIRI HASHIMOTO,[4] HITOSHI IKEDA,[4] AKIHIRO KUSUMI,[5] NOBUAKI MIYAKAWA[6]

[1]*Component Development Department, Suzuka Fuji Xerox Co., Ltd., 2274 Hongo, Ebina, Kanagawa 243-04, Japan*
[2]*Research Center, Taisho Pharmaceutical Co., Ltd., 1-403 Yoshino, Ohmiya, Saitama 330, Japan*
[3]*Department of Mathematics and Computer Science, Shimane University, 1060 Nishikawatsu, Matsue 690, Japan*
[4]*Document Engineering Laboratory, Fuji Xerox Co., Ltd., 430 Sakai, Nakai, Ashigarakami, Kanagawa 259-01, Japan*
[5]*Department of Biological Science, Nagoya University, Chikusa, Nagoya 464-01, Japan*
[6]*Electronic Components Technology Department, Fuji Xerox Co., Ltd., 2274 Hongo, Ebina, Kanagawa 243-04, Japan*

Received 22 September 1997; accepted 28 August 1998

ABSTRACT: Application of molecular dynamics (MD) simulations to large systems, such as biological macromolecules, is severely limited by the availability of computer resources. As the size of the system increases, the number of nonbonded forces (Coulombic and van der Waals interactions) to be evaluated increases as $O(N^2)$, where N is the number of particles in the system. The force evaluation consumes more than 99% of the CPU time in an MD simulation involving over 10,000 particles. Hence, the major target for reduction of the CPU time should be acceleration of the calculation of nonbonded forces. For this purpose, we developed a custom processor for calculating nonbonded interactions and a scalable plug-in machine (to a workstation), the MD Engine, in which numbers of the custom processors work in parallel. The processor has a pipeline architecture to calculate the total nonbonded force using the coordinates, electric charge, and species of each particle broadcast by the host

Correspondence to: Dr. T. Amisaki; e-mail: ami@cis.shimane-u.ac.jp

Journal of Computational Chemistry, Vol. 20, No. 2, 185–199 (1999)

CCC 0192-8651 / 99 / 020185-15

computer. The force is calculated with sufficient accuracy for practical MD simulations. The processor also calculates virials simultaneously with forces for use in the calculation of pressure, accommodates periodic boundary conditions, and can be used in Ewald summations. An MD Engine system consisting of 76 processors calculates nonbonded interactions about 50 times faster than an UltraSPARC-I processor (Sun Ultra-2, 200 MHz) or an R10000 processor (SGI Origin 200, 180 MHz). On a Sun Ultra-2 workstation with a single UltraSPARC-I processor an MD simulation of a Ras p21 protein molecule immersed in a water sphere (13,258 particles) was accelerated by a factor of 48 using the MD Engine system. © 1999 John Wiley & Sons, Inc. J Comput Chem 20: 185–199, 1999

Keywords: special-purpose computer; hardware; performance model; nonbonded interaction; Ewald method

Introduction

In recent years, the dramatic growth of computational power has made molecular dynamics (MD) a practical method to understand the physical properties of biological macromolecules at the atomic level. The method has penetrated into the fields of protein engineering,[1] drug design,[2,3] and refinements of structures based on X-ray[4] and NMR experiments.[5]

However, the use of MD simulations in studies of large biological systems is severely limited by the availability of computer resources. Because the computational complexity of MD calculations is governed by the number of nonbonded (Coulombic and van der Waals) interactions to be evaluated, it increases as $\mathcal{O}(N^2)$ with an increase of N. (N is the number of particles in the system.) One approach to this problem is to use fast algorithms such as the hierarchical method[6] or multipole approximations.[7] Another approach is to produce special-purpose machines such as FASTRUN[8] and GRAPE-2A,[9] which have pipelines of digital signal processors to calculate nonbonded forces.

In the present study we advance the latter approach. By taking advantage of recent cost reductions in the design and production of application-specific integrated circuit (ASIC) chips, it is possible to produce parallel machines that are more efficient than general-purpose parallel machines in terms of calculation time and cost. We developed a processor chip that has an embedded pipeline to calculate nonbonded interactions (called MODEL, representing a molecular dynamics processing element). In the design of MODEL the circuit was optimized for sufficient accuracy with minimal complexity to achieve high cost performance without compromising the accuracy of the simulation. The accuracy of nonbonded forces required for stable MD simulations has been systematically studied by Amisaki et al.[10] and provided the basis for the present work.

The MODEL chip allowed us to produce a special-purpose parallel machine, the MD Engine. The MD Engine is an accelerator interfaced to a host workstation. The machine receives the coordinates, electric charge, and species of each particle in the molecular system from the host computer, then calculates all of the nonbonded forces and sends these values back to the host computer. In this manner, the remaining MD calculation, which is complex but only consumes a very limited percentage of computation time, can be left to the host computer and the ASIC design can be dramatically simplified: the sole task of the MD Engine is to calculate the nonbonded forces acting on a particle from all of the other particles in the system. In addition, the MD Engine calculates virials simultaneously with forces for calculation of pressure and accommodates periodic boundary conditions and Ewald summation.

An MD Engine system consisting of 76 processors calculates nonbonded interactions about 50 times faster than an UltraSPARC-I processor (Sun Ultra-2, 200 MHz, Sun Microsystems) or an R10000 processor (SGI Origin 200, 180 MHz). On a Sun Ultra-2 workstation with a single UltraSPARC-I processor an MD simulation of a Ras p21 protein molecule immersed in a water sphere (13,258 particles) was accelerated by a factor of 48 using the MD Engine system. Furthermore, the MD Engine is highly scalable as shown in the present report, which allows further acceleration with additional MODEL processors.

We first describe the basic concepts involved in development of the MD Engine. Then the architec-

ture of the MD Engine is described, followed by the architecture and function of the MODEL processor. A brief description of a logic for evaluating the forces is also given. The performance of the MD Engine is reported.

Basic Concepts of MD Engine

We first illustrate how the evaluation of nonbonded forces in an MD simulation is very costly in regard to time. The molecular system examined is a protein molecule immersed in water and contains 13,258 particles: 1762 for a Ras p21 protein, 37 for a GTP, 1 for a Mg^{2+} ion, 4 for Na^{+} ions, and 11,454 for a sphere of water molecules. The actual time required for calculation of a single time step of the simulation is summarized in Table I. Simulations were carried out using AMBER 4.1[11] with a cutoff radius of 15 Å for the Lennard–Jones (LJ) interaction. The Coulombic interaction was not truncated. As can be seen, it took several tens of seconds to advance a time step in the simulation on either the Sun Ultra-2 or SGI Origin 200 (nonparallelized results). Most importantly, 99.8% of the simulation time was spent in calculating nonbonded forces. Clearly, acceleration of this portion greatly contributed to the speedup of the total calculation.

The computational complexity of the evaluation of nonbonded forces is high (on the order of N^2) because all pairs have to be considered to obtain the total force. However, the force evaluation consists of a large number of repetitions of simple calculations: the summing of pairwise forces that are simply a function of the distance between particles. In addition, the force exerted on each particle can be evaluated independently. Because of these properties, nonbonded interaction calculations might be suitably addressed by the design of a special-purpose parallel machine.

TABLE I.
Time Required for MD Simulation of Ras p21 Protein–Water System on Workstations.

	Sun Ultra-2		SGI Origin 200	
Part	Time (s)	Percentage	Time (s)	Percentage
Nonbonded[a]	65.82	99.8	59.85	99.8
Other	0.13	0.2	0.13	0.2
Total	65.95	100.0	59.98	100.0

The values are average times for a single time step; nonparallelized results.
[a]Evaluation of Coulombic and van der Waals interactions.

One of the prevalent methods for acceleration of MD simulations is employing cutoffs for the LJ forces and the long-range Coulombic forces. However, cutoffs of the Coulombic interactions cause various detrimental effects and make the simulation unreliable.[12–15] The aim of this work was to provide a fast method for MD simulations without a cutoff of the Coulombic interactions.

Our machine for calculating nonbonded interactions was designed as a hardware accelerator to be plugged into a host computer. The accelerator receives the coordinates, electric charges, and species of particles from the host computer and then calculates the total force and sends it back.

The machine, which we named the MD Engine, is a parallel computer: it can be viewed as a homogeneous array of custom processor elements. The processor is called MODEL, and it contains circuits to carry out all the necessary operations to evaluate nonbonded forces. The remainder of this section specifies the fundamental requirements for a practical MD simulation, all of which are implemented in the MODEL processor.

POTENTIAL AND FORCE FUNCTIONS

In conventional MD simulations of biological systems, the potential for a particle i, Φ_i, is modeled as

$$\Phi_i = \Phi_i^B + \sum_{j \neq i} \varepsilon_{ab} \left\{ \left(\frac{\sigma_{ab}}{|\mathbf{r}_{ji}|} \right)^{12} - \left(\frac{\sigma_{ab}}{|\mathbf{r}_{ji}|} \right)^{6} \right\} + q_i \sum_{j \neq i} \frac{q_j}{|\mathbf{r}_{ji}|}, \quad (1)$$

where $\mathbf{r}_{ji}$ is a vector from the particles j to i, and q_i is the charge of the particle i. The first term Φ_i^B is the bonded potential describing the contributions from bonded particles. The computational complexity of evaluating Φ_i^B is $\mathcal{O}(1)$, because only a few particles are covalently coupled to the ith particle. The second term, which describes van der Waals interaction, is the LJ potential characterized by an energy parameter ε_{ab} and a length parameter σ_{ab}, where a and b denote the two species of particles. With this potential, an LJ force $\mathbf{f}_i^{LJ}$ can be expressed as

$$\mathbf{f}_i^{\mathrm{LJ}} = \sum_{j \neq i} \frac{\varepsilon_{ab}}{\sigma_{ab}^2} \left\{ 12 \left(\frac{\sigma_{ab}}{|\mathbf{r}_{ji}|} \right)^{14} - 6 \left(\frac{\sigma_{ab}}{|\mathbf{r}_{ji}|} \right)^{8} \right\} \mathbf{r}_{ji}. \quad (2)$$

The third term in the right-hand side of eq. (1) is the Coulombic (C) potential, and the correspond-

ing force $\mathbf{f}_i^{\mathrm{C}}$ is expressed as

$$\mathbf{f}_i^{\mathrm{C}} = q_i \sum_{j \neq i} \left(q_j / |\mathbf{r}_{ji}|^3 \right) \mathbf{r}_{ji}. \quad (3)$$

The computational complexity of evaluating these nonbonded potential or force functions is $\mathcal{O}(N)$ for each particle. Hence, $\mathcal{O}(N^2)$ time is required to evaluate the functions for all particles in the system. As mentioned previously, accelerating these evaluations is the prime target for the design of the MD Engine. Note that, although only the LJ and Coulombic interactions are illustrated in this section, the MD Engine can also deal with an arbitrary central potential or force function.

CUTOFF CONVENTION AND NEIGHBOR LIST

The cutoff convention is the simplest and oldest method for reducing the computation time. This method ignores contributions from particles outside a certain cutoff radius r_c and reduces the time complexity to $\mathcal{O}(N)$. However, the Coulombic interaction is long range [i.e., it decreases slowly with an increase of distance (eq. (3)] and ignoring the long-range interactions makes the simulation unreliable.[12–15] Therefore, in the present development, no truncation of the Coulombic force was employed.

In contrast, the LJ force decreases rapidly with increasing distance [eq. (2)]. The interaction can thus be truncated without affecting the results in most cases, provided the particles are well separated (i.e., $|\mathbf{r}_{ji}| \geq r_c$ with an appropriate value of r_c). When calculating a Coulombic force acting on a particle without truncation, the MD Engine generates a neighbor list (a set of particles located within a distance r_c) for that particle. It then uses this list to calculate the van der Waals forces acting on the particle. This procedure dramatically reduces the computation time and has been incorporated into the MODEL processor (but the Coulombic interaction was not truncated). This procedure differs from the usual cutoff or neighbor-list convention in that a list for each particle is constructed on every time step.

PERIODIC BOUNDARY CONDITIONS

MD simulations are often performed under periodic boundary conditions. In the most common case, a sample volume (the original cell) is a rectangular box and it is surrounded by 26 images of the original cell. Pairwise interactions should be calculated in accordance with the minimum image convention. Namely, a force exerted on the particle i from the particle j is only to be calculated for the real particle j or whichever of its images is nearest to the particle i. This selection is carried out as follows: for each direction $p \in \{x, y, z\}$

$$\text{if} \quad r_{p,ji} \leq -L_p/2 \quad \text{then} \quad r_{p,ji} \leftarrow r_{p,ji} + L_p,$$
$$\text{else if} \quad r_{p,ji} \geq L_p/2 \quad \text{then} \quad r_{p,ji} \leftarrow r_{p,ji} - L_p,$$

where $r_{p,ji}$ is the p component of the vector $\mathbf{r}_{ji}$, and L_p is the size of a cell in the p direction. This operation is included in the MODEL processor.

VIRIAL

Pressure can be defined in terms of virials. The virial $\mathbf{v}_i$ on the particle i can be calculated as

$$\mathbf{v}_i = \sum \mathbf{f}_{ji} \mathbf{r}_{ji}^T,$$

where the superscript T denotes the transpose of the vector. The time complexity of this operation is $\mathcal{O}(N^2)$ [$\mathcal{O}(N)$ for each particle]. Of the nine components of each virial, the MODEL processor incorporates the calculation of only the three diagonal components, because calculation of all the components by the MODEL processor would significantly increase the circuit size. The diagonal components are sufficient to calculate the pressure in each of the three directions, which allows regulation of pressure in the three directions independently.

EWALD METHOD

The Ewald method[16] allows precise calculation of Coulombic forces in cases where electrically charged particles exist periodically. In the Ewald method, force $\mathbf{f}_i^{\mathrm{C}}$ is split into the sum of two rapidly converging series: the real space sum, $\mathbf{f}_i^r$, and the reciprocal space sum, $\mathbf{f}_i^m$:

$$\mathbf{f}_i^{\mathrm{C}} = \mathbf{f}_i^r + \mathbf{f}_i^m.$$

When a positive parameter κ is taken to be an appropriate value so that the $\mathbf{f}_i^r$ converges rapidly, the real space sum is given as

$$\mathbf{f}_i^r = q_i \sum_j q_j \left\{ \frac{2\kappa}{\sqrt{\pi}} \exp\left(-\kappa^2 |\mathbf{r}_{ji}|^2 \right) + \mathrm{erfc}\left(\kappa |\mathbf{r}_{ji}| \right) \right\} \frac{1}{|\mathbf{r}_{ji}|^3} \mathbf{r}_{ji}, \quad (4)$$

where erfc is the complementary error function defined as

$$\mathrm{erfc}(x) = 1 - \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2)\, dt.$$

The reciprocal space sum $\mathbf{f}_i^m$ is given as follows:

$$\mathbf{f}_i^m = 2q_i \sum_k \frac{c_k^{(1)} \exp\left(-\pi^2 |\mathbf{h}_k|^2/\kappa^2\right)}{|\mathbf{h}_k|^2} (\sin 2\pi \mathbf{r}_i \cdot \mathbf{h}_k)\mathbf{h}_k$$

$$- 2q_i \sum_k \frac{c_k^{(2)} \exp\left(-\pi^2 |\mathbf{h}_k|^2/\kappa^2\right)}{|\mathbf{h}_k|^2}$$

$$\times (\cos 2\pi \mathbf{r}_i \cdot \mathbf{h}_k)\mathbf{h}_k, \quad (5)$$

$$c_k^{(1)} = \sum_j q_j \cos 2\pi \mathbf{h}_k \cdot \mathbf{r}_j, \quad (6)$$

and

$$c_k^{(1)} = \sum_j q_j \sin 2\pi \mathbf{h}_k \cdot \mathbf{r}_j, \quad (7)$$

where $\mathbf{h}_k$ is the kth reciprocal vector and its three components are integers. Strictly, the vector $\mathbf{h}_k$ should be defined as $\mathbf{h}_k = (n_{k,x}/L_x, n_{k,y}/L_y, n_{k,z}/L_z)$ where $n_{k,x}$, $n_{k,y}$, and $n_{k,z}$ are integers. Without loss of generality, however, we assume hereafter that the sampling cell is a cube with the region size of unity. The summation over the reciprocal space is carried out for all $\mathbf{h}_k$ such that $|\mathbf{h}_k| \le M$, where the value of M is determined in relation to the value of κ. When the values of the pair (M, κ) are taken appropriately, the number of reciprocal lattice vectors is on the same order of the number of particles in the system. Therefore, the computation time to evaluate Eq. (5) is $\mathcal{O}(N^2)$ [$\mathcal{O}(N)$ for each particle]. The MODEL processor can evaluate $\mathbf{f}_i^r$ and $\mathbf{f}_i^m$ according to eqs. (4) and (5), respectively. The details will be described in a later section.

System Architecture of MD Engine

Figure 1 shows a possible configuration of an MD Engine system. The system consists of a host computer (Sun Ultra-2) and up to four MD Engine chassis. Upon reception of the particle coordinates (or reciprocal vectors for Ewald summation) from the host computer, the MD Engine calculates forces and virials [or the coefficients $c_k^{(1)}$ and $c_k^{(2)}$ that occur in eq. (5)] and sends them back to the host computer. An MD Engine chassis may house up to 19 MD Engine cards (23.3 × 16.0 cm 6U Eurocard size). An MD Engine card has a standard Versa Module Europe (VME) interface. On each card, four MODEL chips are placed. Each MODEL chip contains circuits to carry out all the necessary operations for evaluating nonbonded forces.

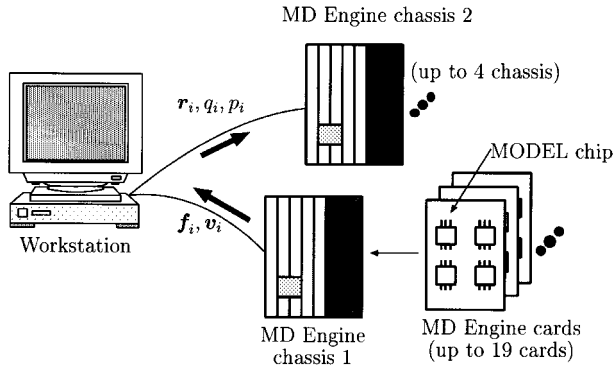

FIGURE 1. The MD Engine system. Up to four MD Engine chassis can be connected to a host computer (workstation), and up to 19 MD Engine cards can be placed in each chassis. The workstation sends the coordinates of particles or reciprocal vectors to the MD Engine and the MD Engine calculates nonbonded forces and virials. The MD Engine then sends these values back to the workstation.

Figure 2 shows the system connection diagram of the MD Engine. It is a single-bus, multiport local memory, multiprocessor system. An interface card connecting the SUN peripheral bus (Sbus) and VME bus is used to mediate communication between the Ultra-2 and the MD Engine. All

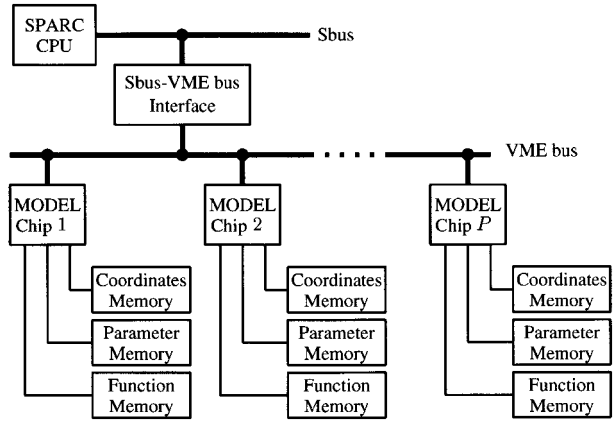

FIGURE 2. A single-bus multiprocessor configuration of the MD Engine. All MODEL chips communicate with a host computer (not shown) through the Sbus–VME bus interface, but they do not communicate with each other. Each chip has its own local memories for storing coordinates, functions, and parameters.

MODEL chips are connected to the VME bus in parallel. This ''single bus multiple processor'' configuration is advantageous in three ways. First, the single bus configuration allows the host computer to directly access the registers and the local memories of MODEL chips. These are mapped onto the memory space of the host computer (memory mapped Sbus devices). Second, the host computer can simultaneously write (broadcast) data on an identical register of every MODEL chip using a global chip address. The cost of this operation does not depend on the number of processors P; in contrast, if it were implemented in software it would require a cost of $\mathcal{O}(\log P)$. Third, the number of MODEL chips can be varied readily: additional MD Engine cards can be plugged into the VME bus slots to increase the computational power as needed. This capacity for reconfiguration also renders the MD Engine ''fault-tolerant'': an MD program may detect newly added or faulty processors during its running time in order to proceed with the maximum number of available processors.

Another important characteristic of the MD Engine is that each MODEL chip has its own local memories: coordinates, function, and parameter memories. These memories contain all the necessary data for calculating a nonbonded force. The coordinates memory stores the coordinates, an electric charge, and an integer representing the species of every particle. The function memory stores the coefficients of the interpolation, which are used to evaluate a function of distance [e.g., $|\mathbf{r}_{ji}|^{-3}$ in eq. (3)]. The parameter memory stores pairs of force field parameters. Due to the architecture, a force evaluation proceeds as follows (actual procedures for the following operation are given in Appendix A): the host computer broadcasts the coordinates of all particles, force field parameters, and coefficients of interpolation to respective local memories of all MODEL chips. Usually, the parameters and the coefficients are broadcast at the start-up time of an MD run and are used throughout the run. Each MODEL chip then carries out the calculation of a nonbonded force using the data resident only on its local memories. This means that no MODEL chip ever accesses the VME bus during its force calculation, and no bus conflicts occur between MODEL chips. Hence, every MODEL chip operates at its maximal speed irrespective of the other processors. Furthermore, the MODEL chip is able to access its three local memories simultaneously, which increases the rate of internal operations.

Design of MODEL Chip

ARCHITECTURE AND FUNCTION OF MODEL CHIP

Figure 3 is the functional block diagram of the MODEL chip. A hardware pipeline that calculates a nonbonded interaction every 400 ns is implemented in a 14.85×14.85 mm^2 silicon chip. The pipeline consists of the six functional units shown. The chip was designed using 0.8 μm CMOS standard-cell technology, and 130,000 gates of transistors are integrated in it. The operating frequency of the chip is 20 MHz, and the power consumption is 2 W. A VME bus slave function is implemented in the chip to reduce the number of electronic parts on the MD Engine card. In this section, we illustrate how the MODEL chip calculates the nonbonded forces. The detailed architectures of various units in the chip are described in Appendix B.

A pairwise Coulombic force between the particles i and j, $\mathbf{f}_{ji}^{C}$, can be written as

$$\mathbf{f}_{ji}^{C} = q_i \cdot q_j \cdot g(s_{ji}) \cdot \mathbf{r}_{ji},$$

where $s_{ji} = |\mathbf{r}_{ji}|^2$ and $g(s) = s^{-3/2}$. As shown in Figure 4A, the pairwise force is calculated inside the MODEL chip according to this equation as follows: unit 1 fetches the coordinates $\mathbf{r}_j$ and the charge q_j of the particle j from the coordinates memory and passes them to the subsequent stages of the pipeline. Unit 2 subtracts $\mathbf{r}_j$ from $\mathbf{r}_i$ to obtain the vector $\mathbf{r}_{ji}$. The coordinates of the particle i are

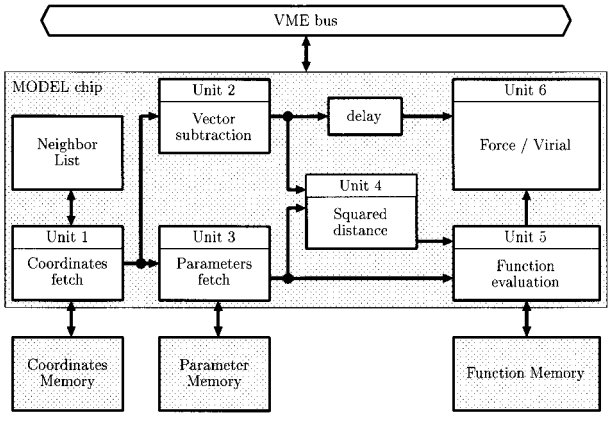

FIGURE 3. A functional block diagram of the MODEL chip. The chip contains a pipeline consisting of six major functional units and calculates nonbonded forces or potentials. Each chip has a VME bus slave function and three external local memories.

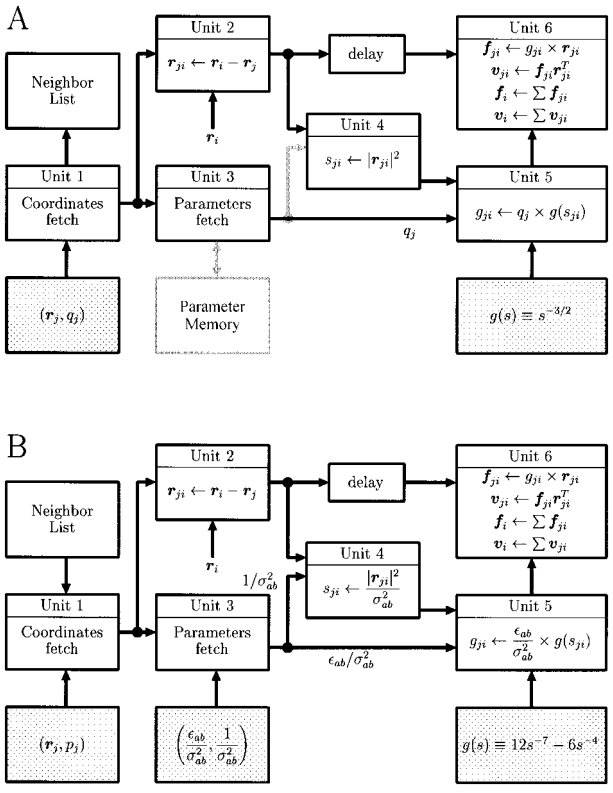

FIGURE 4. The operations for calculating nonbonded forces. (A) Coulombic force. The coordinates and charge are fetched from the external coordinates memory and put into the pipeline that calculates force and virial. A neighbor list is generated simultaneously. (B) The LJ force. Interactions are evaluated only for the particles that are resident on the neighbor list. Force field parameters are fetched from the parameter memory.

obtained through registers inside the MODEL chip. When periodic boundary conditions are applied, the unit automatically selects the minimum image of the particle. Unit 3 is bypassed when calculating a Coulombic force. Unit 4 calculates the squared distance between the particles, s_{ji}. If s_{ji} is smaller than a certain value s_c specified in advance, then the particle j is registered on the neighbor list. In unit 5 the function g is evaluated using the second-order polynomial interpolation. The coefficients of interpolation are retrieved from the function memory. Usually, these coefficients are written on the memory at the start-up of an MD run and are used throughout the run. The memory capacity is sufficient to store coefficients of four different functions. Finally, unit 6 calculates the pairwise force ($\mathbf{f}_{ji}^{\mathrm{C}}/q_i$ instead of $\mathbf{f}_{ji}^{\mathrm{C}}$, strictly speaking), which is added on an accumulator in the same unit to obtain the total Coulombic force acting on the particle i. Virials $\mathbf{v}_i = \sum \mathbf{f}_{ji} \mathbf{r}_{ji}^T$ are also calculated in unit 6.

Similar, but slightly different operations are performed to obtain the LJ force (Fig. 4B). A pairwise LJ force can be written as

$$\mathbf{f}_{ji}^{\mathrm{LJ}} = \frac{\varepsilon_{ab}}{\sigma_{ab}^2} \cdot g(s_{ji}) \cdot \mathbf{r}_{ji},$$

where $g(s) = 12s^{-7} - 6s^{-4}$ and s_{ji} is the scaled squared distance defined as

$$s_{ji} = |\mathbf{r}_{ji}|^2 / \sigma_{ab}^2.$$

In this case, unit 1 fetches the coordinates of the particle j that is resident on the neighbor list; that is, interactions are evaluated only for the particles located inside the sphere of the radius $s_c^{1/2}$ with center at $\mathbf{r}_i$. Unit 3 procures a pair of force field parameters $(\varepsilon_{ab}/\sigma_{ab}^2, 1/\sigma_{ab}^2)$ from the parameter memory, and the parameters are passed to the subsequent stages. Unit 4 calculates the scaled squared distance s_{ji} using one of the parameters. The value is passed to unit 5, where the value of $g(s_{ji})$ is calculated and multiplied with $\varepsilon_{ab}/\sigma_{ab}^2$.

When Coulombic force is calculated by the Ewald method, the MODEL chip is used in multiple steps. In the first step, the real space sum $\mathbf{f}_i^r$ defined in eq. (4) is evaluated in the same way as the ordinary Coulombic force. The distinction is in the differing contents of the function memory. The next step is to calculate the coefficients $c_k^{(1)}$ and $c_k^{(2)}$, which are defined in eqs. (6) and (7), respectively. The coefficient $c_k^{(1)}$, for example, is given as the sum of

$$c_{kj}^{(1)} = q_j \cdot g(s_{kj})$$

for all particles j, where $s_{kj} = \mathbf{h}_k \cdot \mathbf{r}_j$ and $g(s) = \cos(2\pi s)$. To calculate the value of $c_{kj}^{(1)}$, the host computer puts $\mathbf{h}_k$ on the registers inside MODEL. Then, as shown in Figure 5A, unit 1 of the MODEL fetches the coordinates $\mathbf{r}_j$ and the charge q_j of the particle j from the coordinates memory. Units 2 and 3 are bypassed, and unit 4 calculates $s_{kj} = \mathbf{h}_k \cdot \mathbf{r}_j$ rather than $s_{ji} = |\mathbf{r}_{ji}|^2 = \mathbf{r}_{ji} \cdot \mathbf{r}_{ji}$. In unit 5 the value of $q_j \cos(2\pi s_{kj})$ is calculated and added on the accumulator in unit 6 to obtain the value of the coefficient $c_k^{(1)}$. The final step of the Ewald method is to calculate the reciprocal space sum $\mathbf{f}_i^m$ [eq. (5)]. For this purpose (Fig. 5B) the host computer puts the elements of each reciprocal vector $\mathbf{h}_k$ and its associated coefficients $c_k^{(1)}$ or $c_k^{(2)}$ multiplied by $\exp(-\pi^2|\mathbf{h}|^2/\kappa^2)$ on the coordinates memory instead of the coordinates and a charge, respectively. Then, respective terms of eq. (5) are calculated

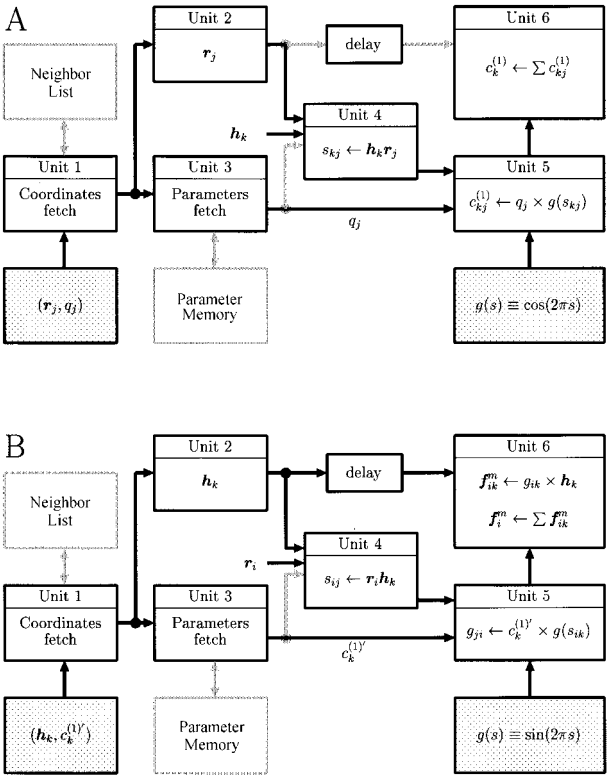

FIGURE 5. The operations for the Ewald summation. (A) Coefficient $c_k^{(1)}$. Dot products $\mathbf{h}_k \cdot \mathbf{r}_j$ are calculated in unit 4. Unit 5 evaluates trigonometric functions. Unit 6 simply performs summation. (B) Reciprocal space sum $\mathbf{f}_i^m$. Reciprocal lattice vectors $\mathbf{h}_k$ and coefficients $c_k^{(1)'}$ are placed on the coordinates memory, where $c_k^{(1)'} = c_k^{(1)} \times \exp(-\pi^2 |\mathbf{h}_k|^2 / \kappa^2)$. The coordinates of the ith particle are placed on a register inside the chip.

using the MODEL chip to obtain the reciprocal space sum $\mathbf{f}_i^m$.

The evaluation of the function g is a key part of the design of the MODEL chip and is described in greater detail in a subsequent subsection.

INTERNAL FORMATS OF NUMBERS

In the design of MODEL, we tried to minimize hardware circuits without compromising the accuracy of calculation. Simplifying the circuitry was necessary to realize high operation speed at low manufacturing cost. The accuracy of calculation is highly dependent on the bit width of data and the rounding algorithm.

Amisaki et al.[10] examined many factors that affect the arithmetic accuracy of a dedicated accelerator that calculates nonbonded interactions and found the following. The pairwise force should be calculated with at least 29 bits of precision using coordinates that, in turn, should have at least 25 bits of precision. The total force acting on a particle, which is the sum of the pairwise forces, should be calculated with at least 48 bits of precision. Calculation of a pairwise force, which involves the quadratic interpolation with a look-up table, requires a key that contains a duplicate of at least 11 of the most significant bits of the mantissa of the squared distance between the particles. (A method described in the next subsection satisfies this requirement.)

In accordance with these findings, we employed two types of precision for floating-point numbers inside the MODEL chip: 40-bit and 64-bit precision. The two are the slightly modified versions of the single extended precision and double precision, respectively, in IEEE standard 754.[17] In our specification, the exponent of the 40-bit number is 8 bits long as shown in Figure 6A. Almost all data (e.g., the coordinates of particles) are handled in the 40-bit number, while accumulation of pairwise forces, as well as virials, are performed in the

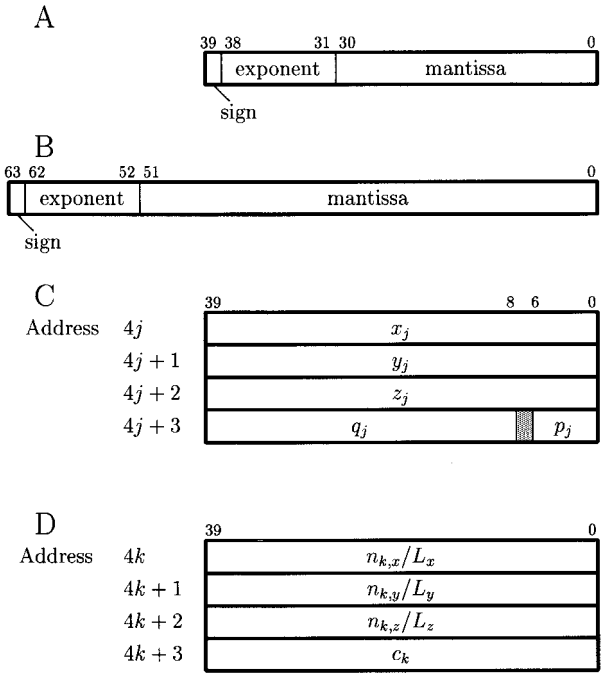

FIGURE 6. The internal formats of numbers. (A) A 40-bit single extended format that is used in the MODEL chip. Almost all data are handled in this format. (B) A 64-bit (double precision) format. Summation of pairwise forces or virial coefficients are done in this format. (C) Layout of the coordinate memory that stores the coordinates (x_j, y_j, z_j), the electric charge q_j, and the 7-bit number p_j that specifies the species of the particle j. (D) Layout of the coordinate memory that stores the reciprocal vectors $(n_{kx}/L_x, n_{ky}/L_y, n_{kz}/L_z)$ and the associated coefficients c_k.

64-bit number (Fig. 6B). In either precision, a denormalized number and NaN (''not a number'') are not supported and every number is rounded toward zero.

As shown in Figure 6C, the coordinates memory of MODEL stores the coordinates $\mathbf{r}_j = (x_j, y_j, z_j)$, the electric charge q_j, and the 7-bit number p_j, which specifies the species of the particle j. A pair of p_j on the coordinates memory and p_i on a register inside MODEL provides a 14-bit address for access to the parameter memory. The format of q_j is IEEE single precision. The format of the force field parameters is also that of IEEE single. Although charges and force-field parameters are all in single precision, their precision is sufficient because they are not changed in an MD simulation. For calculation of Coulombic force using the Ewald method, reciprocal lattice vectors $\mathbf{h}_k = (n_{k,x}/L_x, n_{k,y}/L_y, n_{k,z}/L_z)$, and their coefficients, $c_k^{(1)}$ or $c_k^{(2)}$, are stored in the coordinates memory as shown in Figure 6D. As will be described later, it was empirically confirmed that the above-mentioned precisions were sufficient to obtain a numerical accuracy required for MD simulations.

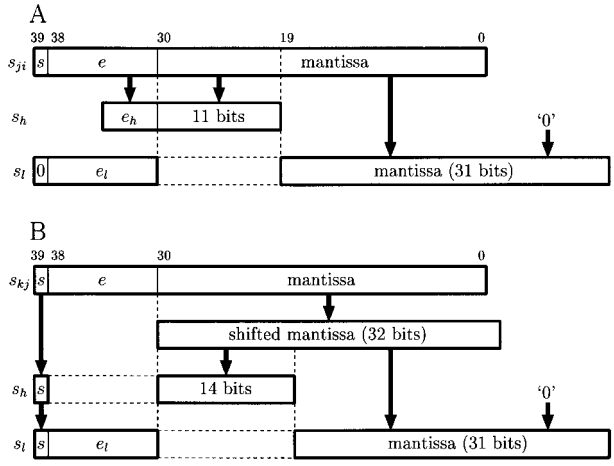

FIGURE 7. The operation required for interpolation with a look-up table. (A) A scaled squared distance s_{ji} is decomposed into the two numbers s_h and s_l. The number s_h is a key to access the function memory and s_l is used in the quadratic interpolation. (B) An argument of the trigonometric functions s_{ji} is decomposed into the two numbers s_h and s_l. The number s_h is a key to access the function memory and s_l is used in the quadratic interpolation.

EVALUATION OF PAIRWISE FORCES

To calculate a pairwise LJ force, MODEL evaluates the function $g(s) = 12s^{-7} - 6s^{-4}$ using the quadratic interpolation. For this purpose, MODEL decomposes the scaled squared distance s_{ji} into two numbers, s_h and s_l, as shown in Figure 7A. The floating-point number s_h represents the scaled squared distance in a range $[2^{-2}, 2^{14})$ using 15 bits: 4 for the exponent and 11 for mantissa. The latter is a copy of the most significant bits of the mantissa of s_{ji}. MODEL calculates the value of $(s_{ji} - s_h) \times 2^{-e+138}$, where e is the exponent of s_{ji}, and normalizes it in a 40-bit floating-point number s_l. The factor 2^{-e+138} is introduced to simplify the hardware. Indeed, MODEL simply stores $(127 - k)$ on e_l, the exponent of s_l, where k is the number of leading zeros in the 20 least significant bits of s_{ji}. Then the function g can be approximated as

$$g(s_{ji}) \simeq (a_2 \times s_l + a_1) \times s_l + a_0,$$

where a_0, a_1, and a_2 are the coefficients of interpolation. A set of the triples of the coefficients (i.e., a look-up table) is stored in the function memory of each MODEL chip. The number s_h is used as a 15-bit address to draw the coefficients a_0, a_1, and a_2. Note that the two coefficients a_1 and a_2 have to be adjusted in advance, taking into account the scaling on s_l.

If the squared distance s_{ji} lies outside the range, MODEL sets the function value to be zero. When $s_{ji} \geq 2^{14}$ this operation is valid, because the LJ force is short range. On the other hand, when $s_{ji} < 2^{-2}$, the particle j is so close to the particle i that the force cannot be precisely calculated using second-order polynomial interpolation. This is because the LJ force varies steeply in this domain. In such a case, the MODEL writes the index number j on the ''nearest neighbor list'' in the internal memory; subsequently, the force should be corrected on the host computer taking into account the particles recorded on the nearest neighbor list. The cost for the correction is low because in most MD simulations only a few particles are recorded in the nearest neighbor list.

For a Coulombic pairwise force, the scaled squared distance s_{ji} is defined as

$$s_{ji} = s_0 |\mathbf{r}_{ji}|^2,$$

where s_0 is a scaling factor stored in the MODEL register named VALS0. As is the case with the LJ force, the function for Coulombic force, $g(s) = s^{-3/2}$, is calculated using the quadratic interpolation. The coefficients are fetched from the location s_h of the function memory. However, the Coulom-

bic force is long range and cannot be considered to have vanished, even if the squared distance is equal to or greater than 2^{14}. In this case we may shift the range of s_h by s_0.

To calculate reciprocal space sums of the Ewald method, evaluation of two trigonometric functions should be established. In this case the function g, which is evaluated using the quadratic interpolation, is the function of s_{kj} defined as $s_{kj} = \mathbf{h}_k \cdot \mathbf{r}_j$. For example, to obtain the coefficient $c_k^{(1)}$ we must evaluate the function $g(s) = \cos(2\pi s)$. Because the function is periodic with an interval of unity, we reduce the range of s_{kj} into $[-1, 1]$; that is, the integer part of s_{ji} is discarded by shifting the mantissa to make the exponent 126. Then the fraction part of s_{kj} is decomposed into two numbers s_h and s_l, where $s_h = \lfloor s_{kj} \times 2^{14} \rfloor \times 2^{-14}$ and $s_l = s_{kj} - s_h$ (Fig. 7B). The value of s_h is represented using a 15-bit integer with a sign bit and is used as an address for accessing the function memory. The value of s_l is normalized into a 40-bit floating-point number to be used in the quadratic interpolation, as is the case with the LJ force function.

Performance of MD Engine

EXPERIMENTALLY OBTAINED PERFORMANCE MEASURES

We first report the numerical accuracy of the MD Engine. One of the most convenient ways to investigate the reliability of an MD simulation is to examine the energy conservation in a constant-energy MD run. We carried out a constant-energy MD simulation of a Ras p21 protein–water system (N = 13,258) with a time step of 2 fs. We used the MD Engine system with 76 MODEL processors connected to a Sun Ultra-2 via an Sbus-VME bus extension adaptor (SVA-100, Aval Data Corp.). In the simulation the root mean square fluctuation of the total energy was small enough, and the total energy was well conserved. The ratio of the fluctuation to the average total energy was 5.4×10^{-5}. The ratio was shown to be a reasonable measure of accuracy of MD simulations[18] with a value of less than about 10^{-3} indicating an acceptable numerical accuracy.

Next we illustrate the performance of the MD Engine in terms of its computational speed. The time required for evaluations of nonbonded forces in a single time step of the MD simulation of the Ras p21 protein system (N = 13,258) was measured and is shown in Table I. In these experiments the LJ interaction was calculated with a cutoff radius of 15 Å, but the Coulombic interaction was calculated without cutoffs (i.e., we modified the source code of AMBER so that a neighbor list was not generated). When the simulation was entirely carried out on a workstation, nonbonded forces were calculated as follows: a distance between a particle pair was calculated to obtain the pairwise Coulombic force, and then the LJ force was calculated using the distance only when it was less than the specified cutoff distance. The MD Engine calculated the total Coulombic force acting on a particle without cutoffs and generated a neighbor list for that particle with no additional costs. The list was then used to calculate the LJ force acting on that particle from the particles located within the cutoff distance.

The computation of the nonbonded forces was completed in 1.25 s using the MD Engine with 76 MODEL chips. The MD Engine system is about 50 times faster than an UltraSPARC-I processor (Sun Ultra-2, 200 MHz) or an R10000 processor (SGI Origin 200, 180 MHz). The overall time required for calculation of a single time step was reduced from 65.95 to 1.38 s on the Ultra-2 (single UltraSPARC-I), which is an acceleration factor of 47.8. The Origin 200 needed 59.98 s for the same calculation, which is 43.5 times slower than the MD Engine system.

PERFORMANCE MODEL

The performance model considered here specifies the execution time for calculating all nonbonded forces required for a single time step of an MD simulation as a function of problem size N, number of MODEL chips P, and other hardware and software characteristics.

During execution each processor is computing, communicating, or idling as illustrated in Figure 8. In this example there are four MODEL chips numbered 1–4, and the scheduling of the calculation of nonbonded forces is made in a round-robin fashion.

The execution time for calculating a total Coulombic force and LJ force are represented as $t_{\text{comp}}^{\text{C}}$ and $t_{\text{comp}}^{\text{LJ}}$, respectively. Because each MODEL chip calculates one pairwise force within 400 ns, $t_{\text{comp}}^{\text{C}}$ and $t_{\text{comp}}^{\text{LJ}}$ are given in microseconds (μs) as

$$t_{\text{comp}}^{\text{C}} = 0.4N$$

and

$$t_{\text{comp}}^{\text{LJ}} = 0.4N_c = 400,$$

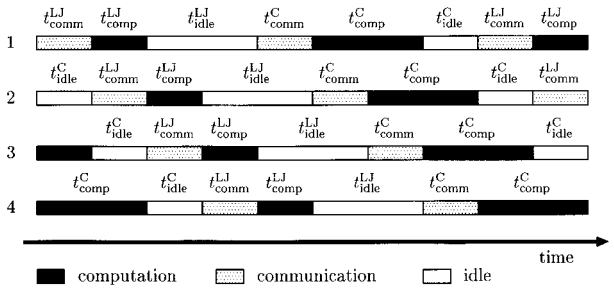

FIGURE 8. A timing chart for MD simulation in the case of $P = 4$. All MODEL chips are working in parallel. The execution time for calculating a total Coulombic force and a total LJ force are represented as t^{C}_{comp} and t^{LJ}_{comp}, respectively. The time t^{C}_{comm} preceding each evaluation of the Coulombic force is the communication time within which the host computer receives the calculated van der Waals force and the associated nearest neighbor list and then sends the data necessary for starting the evaluation of the Coulombic force. The communication time t^{LJ}_{comm} is that preceding the evaluation of the LJ force. The time t^{C}_{idle} and t^{LJ}_{idle} drawn with open boxes represents idle time during which the MODEL chips are waiting to send or receive data.

where N_c is the number of particles in a neighbor list. For simplicity, we assume $N_c = 1000$ in this analysis.

The durations t^{C}_{comm} and t^{LJ}_{comm} are the time spent mainly in communication between each processor and the host computer. Within t^{C}_{comm}, for example, the host computer receives the calculated LJ force and the associated nearest neighbor list, then sends the data required to start the evaluation of the Coulombic force. The communication times t^{C}_{comm} and t^{LJ}_{comm} are estimated as about $30t_{comm}$ each, where t_{comm} is a unit time for a 32-bit data transmission through the VME bus and is measured in this work as about 0.7 μs.

The times t^{C}_{idle} and t^{LJ}_{idle} are idle times during which the MODEL chips are waiting for receiving or sending data. These are given as

$$t^{C}_{idle} = \max\left\{(P-1)t^{C}_{comm} - t^{C}_{comp}, 0\right\}$$

and

$$t^{LJ}_{idle} = \max\left\{(P-1)t^{LJ}_{comm} - t^{LJ}_{comp}, 0\right\}.$$

Consequently, the total time for evaluating nonbonded forces, T, can be approximated as

$$T = \left(t^{C}_{comm} + t^{C}_{comp} + t^{C}_{idle} + t^{LJ}_{comm} + t^{LJ}_{comp} + t^{LJ}_{idle}\right)N/P + t^{B}_{comm}, \quad (8)$$

where $t^{B}_{comm} = 8Nt_{comm}$ (not shown in Fig. 8) is the time for broadcasting the coordinates, charges, and species of all particles to the coordinates memory.

As previously described, evaluation of nonbonded forces in the MD simulation of the Ras p21 molecular system (N = 13,258) required 1.25 s with the MD Engine system (76 processors). According to the performance model, the expected time is 1.28 s, which agrees well with the observed value.

Figure 9 plots the expected total execution time T against the number of particles N. The solid lines show the performance model, eq. (8). The dotted line indicates the case where neither t^{C}_{idle} nor t^{LJ}_{idle} is zero, as is true for the example in Figure 8. In this case, the total execution time is limited by the communication between the host computer and the MODEL chips, is independent of P, and is proportional to N. In other words, in the region where the solid lines lie on the dotted line, the performance of the MD Engine is saturated, and the execution time cannot be reduced even if additional MODEL chips are used. By contrast, when the number of particles are relatively large as compared with the number of MODEL chips, a linear increase in calculation rate is obtained.

In our performance analysis, we assume that the calculation of nonbonded forces and the other computations required for a single time step of the

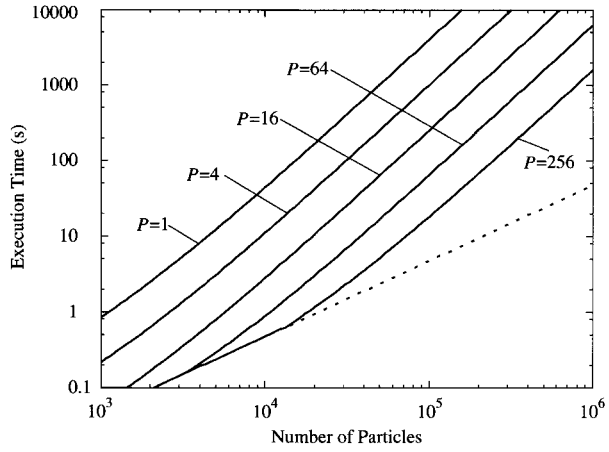

FIGURE 9. The expected calculation time for an MD Engine with P MODEL chips to calculate all nonbonded forces required for a single time step of MD calculation as a function of the number of particles in the system, N. The dashed line indicates the case wherein N is too small, and the computation time is limited by the performance of communication between the host computer and MODEL chips. In other regions where N is relatively large, linear speedup is attained.

MD run are performed in series, which was the case with the actual MD run presented in this article. If these two sets of computations were to be performed in parallel, the simulation time could be further reduced.

The MD Engine can be viewed as an implementation of a simple replicated data algorithm, which is characterized by a design in which each processor stores its own copy of the simulation data (e.g., the coordinates of the particles). The domain decomposition strategy is another parallel algorithm and is generally more efficient than the replicated data method.[19,20] Both parallel algorithms are intended to be used for short-range forces. High scalability for long-range interactions is difficult to achieve.[19] It is worthwhile to note that, according to the performance model, the MD Engine shows a high efficiency for long-range Coulombic forces. This is mainly due to its single-bus multiple processor configuration.

Concluding Remarks

We developed a special-purpose parallel machine, the MD Engine, for use in MD simulations. It is plugged into a workstation as an accelerator for calculation of nonbonded forces, which is the most time-consuming part of MD simulations. In addition, it calculates virials simultaneously with the forces and accommodates the periodic boundary conditions and the Ewald method. An accuracy in computations sufficient to carry out normal MD simulations was achieved.

One of the remarkable characteristics of the MD Engine is its high scalability at a practical level: the MD Engine's computational power is increased linearly with the number of MODEL processors. Other notable characteristics of the MD Engine are that it is fault tolerant and reconfigurable with regard to the number of its constituent processors.

Using an MD Engine system composed of 76 processors, the MD simulation of the Ras p21 molecular system (N = 13,258) was accelerated by a factor of 47.8 on the Sun Ultra-2 workstation. It is 43.5 times faster than an SGI Origin 200 (single R10000 processor). These figures could be further improved by employing additional MODEL chips and/or parallelizing the calculations on the MD Engine and a host computer.

The web page of AMBER[21] presents benchmark results on the latest computers when applied to the MD simulation of a solvated protein molecule. The simulation is similar to ours but differs mainly in that the Coulombic interactions were truncated at 12 Å. Although the results cannot be directly compared with those presented in this article, we can obtain rough estimates of relative performances of those machines for a practical MD simulation of biological molecules. According to the reports the Cray T90 (single processor), Cray T90 (four processors), Cray C90 (single processor), Hitachi SR2201 (64 processors), and Intel Paragon (128 processors) are about 3, 10, 2, 20 and 5 times as fast as the SGI Origin 200 (single processor), respectively. The performance of the MD Engine is thus better than the latest supercomputers with moderate scales.

The MD Engine does not improve the asymptotic complexity of MD simulations, which remains at $\mathcal{O}(N^2)$. For an extremely large molecular system (e.g., $N > 10^6$), we have to reduce the complexity using, for example, the fast multipole method (FMM)[7] and multiple time step methods such as r-RESPA.[22] The accuracy of FMM can be considerably improved by increasing the number of pairwise forces evaluated directly.[23,24] Therefore, a combination of the approaches with the MD Engine may be useful in the future, although the current design of the MODEL chip is not suitable for such a combination. A new machine that is suitable for such calculations is now being developed in our laboratories.

Appendix A: Programming for MD Engine

The software for the MD Engine system consists of three classes: a low-level library, an interface library, and an application program. The low-level library is a set of C language functions that are called from the interface library. The library contains a function that initializes the MODEL chips, a function that establishes the mapping between the address space of the application program and Sbus devices, and a function that enables access to the registers of the MODEL chips. For example, the following statement, written in C language, is a function call to store data within a register inside the MODEL chip.

```
mdeWrite(pid, regid, value)
```

Here, data, the value of which is given by `value`, is stored on the register `regid` inside the MODEL chip having an ID number of `pid`. When the data

are broadcast to all MODEL chips, a special value, ALL−CHIPS, is specified in place of the particular ID number of a MODEL chip.

The interface library is a set of C or FORTRAN language functions that are called from application software. The library hides the hardware details of the MD Engine from application programmers. For example, the function

```
mdeInstallPairWiseFunc(pid, tid, func)
```

generates a look-up table for the quadratic interpolation and transfers it to the function memory. The value of tid indicates the number of the table (0−3). The function to be approximated is specified with the pointer func, and it may be LennardJones-Force defined below.

```
double LennardJonesForce(double s)
{
  return 12.0 * pow(s, -7.0)
    - 6.0 * pow(s, -4.0);
}
```

The function will be called with the scaled squared distance s_{ji}.

By using the interface library, the development or modification of an application program becomes an easy task. At present, a patch for the source codes of AMBER 4.1 is available on request.

Appendix B: Detailed Architecture of Functional Units

This section describes the six major functional units of the MODEL chip (Fig. 3).

UNIT 1: COORDINATES FETCH

Figure B1 shows the functional unit. The coordinates memory is a 512K × 40-bit SRAM. ADCORD is a 19-bit address counter, the 17 most significant bits of which specify the index number of a particle. Thus, up to $2^{17} = 131{,}072$ particles can be stored in the memory at once. A larger molecular system can be handled by decomposing the system. The least two significant bits of ADCORD are used to specify the kind of data (i.e., x, y, or z coordinates or a charge q). The neighbor list may be stored in NBRM, which is a 2048 × 17-bit internal SRAM, and up to 2048 particles can be recorded on it.

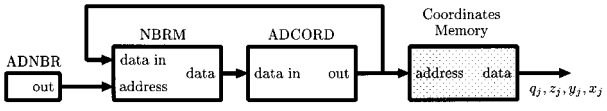

FIGURE B1. Configuration of unit 1.

The calculation of Coulombic interactions is initiated by writing $(N-1) \times 4$ on ADCORD. The unit then fetches the coordinates and the charge of a particle within four clock cycles (400 ns). When calculating with the aid of the Ewald method, the reciprocal lattice vector with the associated coefficient is fetched in a similar way.

A neighbor list may be generated as follows: if a square distance calculated in unit 3 is less than a certain cutoff value s_c, which has to be set in a register inside the MODEL chip before starting the calculation, the index number of the particle is registered on NBRM and the value of ADNBR is incremented by one. The generated neighbor list may be used in calculating the LJ force. In this case, the calculation is started by setting ADNBR to be $N_c - 1$. The index number of a particle is fetched from NBRM to be written on ADCORD. Then the coordinates are fetched from the coordinates memory, and, at the same time, the content of ADNBR is decremented and the next index number is fetched.

UNIT 2: VECTOR SUBTRACTION

Figure B2 shows the block diagram of the functional unit for calculating a relative position vector. The floating-point subtracter $(A - B)$ calculates $\mathbf{r}_{ij} = \mathbf{r}_j - \mathbf{r}_i$. If periodic boundary conditions are applied, the comparator and the floating-point arithmetic unit $(A \pm B)$ corrects the vector for the condition as described in the text. Otherwise, input B for the arithmetic unit $(A \pm B)$ is set to zero.

UNIT 3: PARAMETERS FETCH

Pairs of force-field parameters $(\epsilon_{ab}/\sigma_{ab}^2, 1/\sigma_{ab}^2)$ are stored on the parameter memory, which is a

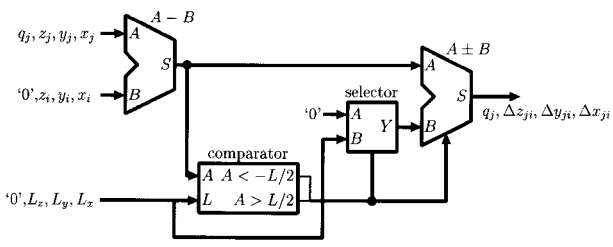

FIGURE B2. Configuration of unit 2.

128K × 16-bit SRAM. A pair is fetched from the memory using the species of the jth and ith particles, p_j and p_i, as an address. On the memory, values of the two parameters are split into two 16-bit data parcels each and restored into original 32-bit numbers in the chip. Usually the memory is capable of storing two sets of 128 × 128 pairs. If the number of the species is less than 65, the memory can be used as eight sets of 64 × 64 pairs of force-field parameters.

UNIT 4: SQUARED DISTANCE

Figure B3 shows the functional block diagram of the unit. When calculating the LJ force, a scaled squared distance

$$s_{ji} = |\mathbf{r}_{ji}|^2 \times 1/\sigma_{ab}^2$$

is calculated in this unit. For the Coulombic force, a scaled squared distance

$$s_{ji} = |\mathbf{r}_{ji}|^2 \times s_0$$

is calculated, where the value of s_0 is stored on the register VALS0. For the Ewald summation, a dot product

$$s_{kj} = \mathbf{h}_k \cdot \mathbf{r}_j$$

is calculated instead. The contents of $\mathbf{h}_k = (n_{k,x}/L_x, n_{k,y}/L_y, n_{k,z}/L_z)$ are supplied from a register inside the MODEL chip.

UNIT 5: FUNCTION EVALUATION

This unit evaluates the function $g(s)$ using the second-order polynomial interpolation. The function can be any pair potential or force or trigonometric functions. The schematic design of the unit is shown in Figure B4. The two numbers s_h and s_l are generated from s_{ji}. The latter is a 40-bit floating-point number to be used in the interpolation. A 15-bit number s_h is used as a part of the address for accessing the function memory, where the three

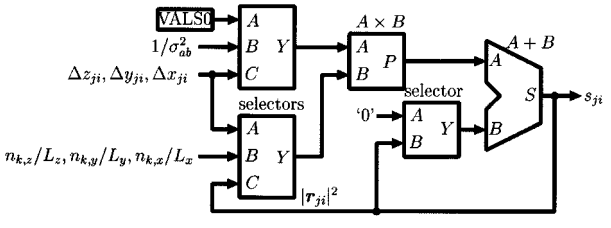

FIGURE B3. Configuration of unit 4.

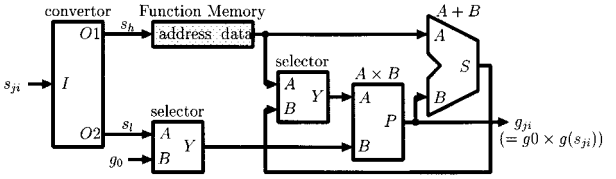

FIGURE B4. Configuration of unit 5.

coefficients of the interpolation are stored for each s_h. The memory is a 512K × 32-bit SRAM with an address width of 19 bits. The two most significant bits of the address are used to select one out of four tables (functions) stored in the memory. The two least significant bits count up automatically to fetch four 32-bit data parcels. The four are converted into three 40-bit floating-point numbers a_0, a_1, and a_2. Using these coefficients, this unit calculates g_{ji} as

$$g_{ji} = ((a_2 \times s_l + a_1) \times s_l + a_0) \times g_0,$$

where g_0 is $\varepsilon_{ab}/\sigma_{ab}^2$, q_j, or c_k.

UNIT 6: FORCE AND VIRIAL

Figure B5 shows a schematic diagram for this unit, which calculates a pairwise force and virial. The vector $\mathbf{r}_{ji}$ is multiplied with the value g_{ji} to obtain a pairwise force $\mathbf{f}_{ji}$. The elements of the pairwise force are multiplied with the respective elements of $\mathbf{r}_{ji}$ to obtain a pairwise virial $\mathbf{v}_{ji}$. Selector 1 may select the input B to calculate potential rather than force. For force calculation, selector 1 selects the input A, three components of $\mathbf{r}_{ji}$ (Δx_{ji}, Δy_{ji}, and Δz_{ji}) are put into the input A with an interval of 100 ns (one clock cycle), and selector 2 switches between its two inputs A and B every 50 ns. This operation produces one element each for force and virial within a single clock cycle.

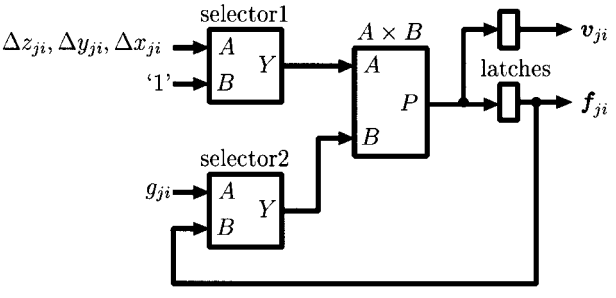

FIGURE B5. Configuration of unit 6 (a part for pairwise interactions).

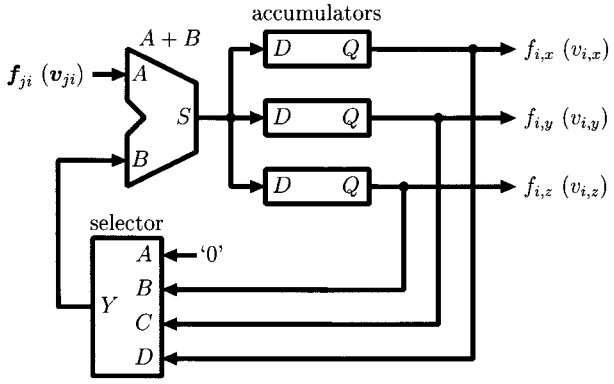

FIGURE B6. Configuration of unit 6 (a part for accumulation).

Total force $\mathbf{f}_i$ is calculated by summing up pairwise forces $\mathbf{f}_{ji}$ using a double precision adder as shown in Figure B6. The total virial $\mathbf{v}_i$ is also calculated on another double precision adder.

Acknowledgments

We thank Mr. K. Hayashi and Ms. A. Araki at Fuji Xerox for their help in designing the MODEL chip, Mr. T. Shimada at Fuji Xerox for his help in assembling a library program, and Mr. K. Uwatoko at Fuji Xerox for his assistance with the MD Engine board reliability testing.

References

1. Prediction of Protein Structure and the Principals of Protein Conformations; Fasman, C. D., Ed.; Plenum Press: New York, 1989.
2. Wasserman, Z. R.; Hodge, C. N. Proteins 1996, 24, 227.
3. Taylor, N. R.; Itzstein, M. J Comput Aided Mol Design 1996, 10, 233.
4. Liao, D.-I.; Silverton, E.; Seok, Y.-J.; Lee, B. R.; Peterkofsky, A.; Davies, D. R. Structure 1996, 4, 861.
5. Greenbaum, N. L.; Radhakrishnan, I.; Patel, D. J.; Hirsh, D. Structure 1996, 4, 725.
6. Barnes, J. E.; Hut, P. Nature 1986, 324, 446.
7. Greengard, L.; Rokhlin, V. J Comput Phys 1987, 73, 325.
8. Fine, R.; Dimmler, G.; Levinthal, C. Proteins 1991, 11, 242.
9. Ito, T.; Fukushige, T.; Makino, J.; Ebisuzaki, T.; Okumura, S. K.; Sugimoto, D.; Miyagawa, H.; Kitamura, K. Proteins 1994, 20, 139.
10. Amisaki, T.; Fujiwara, T.; Kusumi, A.; Miyagawa, H.; Kitamura, K. J Comput Chem 1995, 16, 1120.
11. Pearlman, D. A.; Case, D. A.; Caldwell, J. W.; Ross, W. S.; Cheatham, T. E. III; Ferguson, D. M.; Seibel, G. L.; Singh, U. C.; Weiner, P. K.; Kollman, P. A. AMBER 4.1; University of California: San Francisco, 1995.
12. Loncharich, R. J.; Brooks, B. R. Proteins 1989, 6, 32.
13. Schreiber, H.; Steinhauser, O. Biochemistry 1992, 31, 5856.
14. Saito, M. J Chem Phys 1994, 101, 4055.
15. Oda, K.; Miyagawa, H.; Kitamura, K. Mol Simul 1996, 16, 167.
16. Rapaport, D. C. The Art of Molecular Dynamics Simulation; Cambridge University Press: New York, 1995; p 263.
17. IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Std. 754-1985, IEEE: New York, 1985.
18. Watanabe, M.; Karplus, M. J Chem Phys 1993, 99, 8063.
19. Plimpton, S.; Hendrickson, B. In Parallel Computing in Computational Chemistry; Mattson, T. G., Ed.; American Chemical Society: Washington, DC, 1995; p 114.
20. Wilson, M. R.; Allen, M. P.; Warren, M. A.; Sauron, A.; Smith, W. J Comput Chem 1997, 18, 478.
21. The web address for the AMBER benchmark results is http://www.amber.ucsf.edu/amber/bench.html.
22. Tuckerman, M.; Berne, B. J.; Martyna, G. J. J Chem Phys 1992, 97, 1990.
23. Bishop, T. C.; Skeel, R. D.; Schulten, K. J Comput Chem 1977, 18, 1785.
24. Shimada, J.; Kaneko, H.; Takada, T. J Comput Chem 1994, 15, 28.